```
RRRRRRRRRRR      MMM          MMM      SSSSSSSSSSS
RRRRRRRRRRRR     MMM          MMM      SSSSSSSSSSSS
RRRRRRRRRRRR     MMM          MMM      SSSSSSSSSSSS
RRR          RRR MMMMMMM    MMMMMMM   SSS
RRR          RRR MMMMMMM    MMMMMMM   SSS
RRR          RRR MMMMMMM    MMMMMMM   SSS
RRR          RRR MMM   MMM   MMM   MMM SSS
RRR          RRR MMM   MMM   MMM   MMM SSS
RRR          RRR MMM   MMM   MMM   MMM SSS
RRRRRRRRRRR      MMM          MMM      SSSSSSSSS
RRRRRRRRRRRR     MMM          MMM      SSSSSSSSS
RRRRRRRRRRRR     MMM          MMM      SSSSSSSSS
RRR    RRR       MMM          MMM               SSS
RRR    RRR       MMM          MMM               SSS
RRR    RRR       MMM          MMM               SSS
RRR       RRR    MMM          MMM               SSS
RRR       RRR    MMM          MMM               SSS
RRR       RRR    MMM          MMM               SSS
RRR          RRR MMM          MMM  SSSSSSSSSSS
RRR       RRR    MMM          MMM               SSS
RRR          RRR MMM          MMM  SSSSSSSSSSS
RRR          RRR MMM          MMM  SSSSSSSSSSS
RRR          RRR MMM          MMM  SSSSSSSSSSS
```

```
RRRRRRRR    MM      MM    333333    FFFFFFFFFF    AAAAAA      CCCCCCCC  EEEEEEEEEE
RRRRRRRR    MM      MM    333333    FFFFFFFFFF    AAAAAA      CCCCCCCC  EEEEEEEEEE
RR      RR  MMMM  MMMM  33      33  FF            AA    AA    CC        EE
RR      RR  MMMM  MMMM  33      33  FF            AA    AA    CC        EE
RR      RR  MM  MM  MM        33    FF            AA    AA    CC        EE
RR      RR  MM  MM  MM        33    FF            AA    AA    CC        EE
RRRRRRRR    MM      MM      33      FFFFFFFF      AA    AA    CC        EEEEEEEE
RRRRRRRR    MM      MM      33      FFFFFFFF      AA    AA    CC        EEEEEEEE
RR  RR      MM      MM        33    FF            AAAAAAAAAA  CC        EE
RR  RR      MM      MM        33    FF            AAAAAAAAAA  CC        EE
RR    RR    MM      MM  33    33    FF            AA    AA    CC        EE              ....
RR    RR    MM      MM  33    33    FF            AA    AA    CC        EE              ....
RR      RR  MM      MM    333333    FF            AA    AA      CCCCCCCC  EEEEEEEEEE    ....
RR      RR  MM      MM    333333    FF            AA    AA      CCCCCCCC  EEEEEEEEEE    ....

LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II        SSSSSS
LL              II        SSSSSS
LL              II              SS
LL              II              SS
LL              II              SS
LL              II              SS
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

```
0000     1         $BEGIN  RM3FACE,000,RM$RMSOFACE,<INTERFACE BETWEEN BLISS AND MARS>,-
0000     2                                                 <NOWRT,PIC,LONG>
0000     3
0000     4 ;
0000     5 ;*****************************************************************************
0000     6 ;*                                                                           *
0000     7 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     8 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     9 ;*  ALL RIGHTS RESERVED.                                                     *
0000    10 ;*                                                                           *
0000    11 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    12 ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    13 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0000    14 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000    15 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
0000    16 ;*  TRANSFERRED.                                                             *
0000    17 ;*                                                                           *
0000    18 ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0000    19 ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    20 ;*  CORPORATION.                                                             *
0000    21 ;*                                                                           *
0000    22 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0000    23 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    24 ;*                                                                           *
0000    25 ;*                                                                           *
0000    26 ;*****************************************************************************
0000    27 ;
0000    28
0000    29 ;++
0000    30 ;
0000    31 ; Facility:     RMS32 INDEX SEQUENTIAL FILE ORGANIZATION
0000    32 ;
0000    33 ; Abstract:
0000    34 ;
0000    35 ;               This is a bunch of interface routines between
0000    36 ;               rms-32 written in mars and rms-32 isam in bliss.
0000    37 ;
0000    38 ; Environment:
0000    39 ;
0000    40 ;               VAX/VMS OPERATING SYSTEM
0000    41 ;
0000    42 ;
0000    43 ; Author: W. Koenig     creation date: 4-APR-1978
0000    44 ;
0000    45 ; MODIFIED BY:
0000    46 ;
0000    47 ;       V03-025 JWT0178         Jim Teague              22-Jun-1984
0000    48 ;               Check for UFO in RM$RND_DEV to avoid prologue
0000    49 ;               processing.
0000    50 ;
0000    51 ;       V03-024 DAS0001         David Solomon           25-Mar-1984
0000    52 ;               Fix broken branches.
0000    53 ;
0000    54 ;       V03-023 DGB0017         Donald G. Blair         02-Mar-1984
0000    55 ;               Allocate full-length FIB to support access mode protected
0000    56 ;               files.
0000    57 ;
```

```
0000     58 ;       V03-022 SHZ0001         Stephen H. Zalewski     06-Feb-1983
0000     59 ;               Modify RM$RND_DEV to return false if device is either
0000     60 ;               foreign or non random access (tape).  This prevents copy from
0000     61 ;               attempting to read the prologue from a foreign disk (bugcheck).
0000     62 ;
0000     63 ;       V03-021 MCN0001         Maria del C. Nasr       28-Mar-1983
0000     64 ;               Modify RM$IALL03, RM$IKEY03, and RM$ISUM03 routines to
0000     65 ;               push parameters in the stack to use general linkages.
0000     66 ;
0000     67 ;       V03-020 RAS0133         Ron Schaefer            16-Mar-1983
0000     68 ;               Add $XABCXRDEF macro to fix up RAS0132.
0000     69 ;
0000     70 ;       V03-019 RAS0132         Ron Schaefer            15-Mar-1983
0000     71 ;               Change $RMSRDEF references to $RJRDEF.
0000     72 ;               Make XCONN03_ARGS and XABCRE_ARGS local.
0000     73 ;               Fix bug in XABCRE_ARGS by adding end of table marker.
0000     74 ;
0000     75 ;       V03-018 TMK0004         Todd M. Katz            24-Feb-1983
0000     76 ;               IRB$V_UPDATE is not being cleared on subsequent $GET/$FINDs.
0000     77 ;               Make sure this bit is cleared within RM$GET3 and RM$FIND3.
0000     78 ;
0000     79 ;       V03-017 RAS0124         Ron Schaefer            15-Feb-1983
0000     80 ;               Change MCN0007 to ignore RM$XAB_SCAN for $CREATE on
0000     81 ;               record-oriented devices opened for block I/O.
0000     82 ;               This prevents a subsequent bugcheck trying to access the prolog.
0000     83 ;
0000     84 ;       V03-016 TMK0003         Todd M. Katz            08-Jan-1983
0000     85 ;               Add support for Recovery Unit Journalling and RU ROLLBACK
0000     86 ;               Recovery of ISAM Files. Modifications have been made to
0000     87 ;               RM$FIND3, RM3UPDATE3, RM3PUT3, and RM$DELETE3. These
0000     88 ;               modifications are basically all the same. Within these routines
0000     89 ;               RMS first checks to see whether this operation is taking place
0000     90 ;               as part of the ROLLBACK Recovery of a prematurely terminated
0000     91 ;               or aborted Recovery Unit. If so, then then in each case the
0000     92 ;               ISAM RU ROLLBACK routine, RM$RU_RECOVER is called with the
0000     93 ;               operation as the sole input parameter. When control returns,
0000     94 ;               the recovery operation completes by calling RM$EXRMS.
0000     95 ;
0000     96 ;       V03-015 RAS0118         Ron Schaefer            18-Jan-1983
0000     97 ;               Change MCN0007 to ignore errors returned by RM$XAB_SCAN
0000     98 ;               on the implicit $DISPLAY from $CREATE.
0000     99 ;
0000    100 ;       V03-014 TMK0002         Todd M. Katz            12-Jan-1983
0000    101 ;               Make a change to RM$DELETE3. Change a CLRL (of IRB$W_UDR_ID) to
0000    102 ;               a CLRW.
0000    103 ;
0000    104 ;       V03-013 RAS0109         Ron Schaefer            15-Dec-1982
0000    105 ;               Add $XABxxxDEF references needed by MCN0007.
0000    106 ;
0000    107 ;       V03-012 LJA0044         Laurie J. Anderson       7-Dec-1982
0000    108 ;               Deallocate the key buffers and related buffer and call
0000    109 ;               rm$discommon (to deallocate the IRAB) on restart error.
0000    110 ;
0000    111 ;       V03-011 MCN0009         Maria del C. Nasr       02-Dec-1982
0000    112 ;               Add routine RM$RND_DEV to be called by RM$CREATE3B.
0000    113 ;               It determines if the device where a file is being created
0000    114 ;               is disk or tape.  Do not do display after creating
```

```
0000   115 ;                   indexed file in magtape since no prologue is written.
0000   116 ;
0000   117 ;       V03-010 MCN0008         Maria del C. Nasr       01-Dec-1982
0000   118 ;               In routine RM$EXTEND3, calculate total allocation for
0000   119 ;               given area and store in AREA$L_TOTAL_ALLOC.
0000   120 ;
0000   121 ;       V03-009 MCN0007         Maria del C. Nasr       01-Dec-1982
0000   122 ;               Add an implicit display of the key, allocation, and
0000   123 ;               summary XABs after $CREATE.
0000   124 ;
0000   125 ;       V03-008 LJA0039         Laurie J. Anderson       8-Nov-1982
0000   126 ;               Add a call to XAB_SCAN for restoring NRP for ISAM files.
0000   127 ;               Also, while in here ... alphabetized the modules for easier
0000   128 ;               finding.  Took out expand macro directive for $AREADEF
0000   129 ;
0000   130 ;       V03-007 KBT0165         Keith B. Thompson       23-Aug-1982
0000   131 ;               Reorganize psects
0000   132 ;
0000   133 ;       V03-006 KBT0113         Keith B. Thompson       6-Aug-1982
0000   134 ;               Remove ref. to upd_sifb_eof
0000   135 ;
0000   136 ;       V03-005 LJA0014         Laurie Anderson         29-Jul-1982
0000   137 ;               Fix a truncation error.
0000   138 ;
0000   139 ;       V03-004 KBT0104         Keith B. Thompson       19-Jul-1982
0000   140 ;               Fix another broken branch
0000   141 ;
0000   142 ;       V03-003 KBT0093         Keith B. Thompson       14-Jul-1982
0000   143 ;               Fix broken branch
0000   144 ;
0000   145 ;       V03-002 TMK0001         Todd M. Katz            17-Jun-1982
0000   146 ;               Implement the cluster RMS next record context solution.
0000   147 ;
0000   148 ;               When a $REWIND is done, as the NRP structure has been removed,
0000   149 ;               zero out the next record positioning fields (IRB$L_CUR_VBN,
0000   150 ;               IRB$W_CUR_ID, IRB$W_CUR_COUNT, IRB$W_POS_ID, IRB$L_POS_VBN,
0000   151 ;               IRB$L_SIDR_VBN, IRB$L_UDR_VBN, IRB$W_UDR_ID) in the IRAB, and
0000   152 ;               set the current key of reference (for next record positioning)
0000   153 ;               field also in the IRAB (IRB$B_CUR_KREF) to that which is
0000   154 ;               specified in the RAB. Also clear the current record and
0000   155 ;               end-of-file bits.
0000   156 ;
0000   157 ;               During a $UPDATE/$PUT, the subfields IRB$L_PUTUP_VBN and
0000   158 ;               IRB$W_PUTUP_ID are used to retrieve the RFA of the record
0000   159 ;               just put or updated for use in unlocking the record. In
0000   160 ;               addition it will only be necessary to unlock this primary
0000   161 ;               data record if the bit IRAB$V_UNLOCK_RP is set. What is
0000   162 ;               important is that the internal current record, and thus the
0000   163 ;               next record does not change because of the $PUT or $UPDATE,
0000   164 ;               regardless of the access mode in which it is done.
0000   165 ;
0000   166 ;       V03-001 LJA0007         Laurie Anderson         25-Mar-1982
0000   167 ;               Change KBUFSZ to reference a macro when computing buffer
0000   168 ;               size and make IFB$B_KBUFSZ a word, now: IFB$W_KBUFSZ.
0000   169 ;
0000   170 ;--
0000   171
```

```
                     0000   172 ;
                     0000   173 ; Include Files:
                     0000   174 ;
                     0000   175 ;       [11,20]lib.mlb and [201,10]r.mlb
                     0000   176 ;
                     0000   177 ; Macros:
                     0000   178 ;
                     0000   179
                     0000   180         $RMSDEF
                     0000   181         $RJRDEF
                     0000   182         $DEVDEF
                     0000   183         $BDBDEF
                     0000   184         $RLSDEF
                     0000   185         $RABDEF
                     0000   186         $IRBDEF
                     0000   187         $IFBDEF
                     0000   188         $FIBDEF
                     0000   189         $FABDEF
                     0000   190         $AREADEF
                     0000   191         $XABDEF
                     0000   192         $XABALLDEF
                     0000   193         $XABCXRDEF
                     0000   194         $XABKEYDEF
                     0000   195         $XABSUMDEF
                     0000   196
                     0000   197 ;
                     0000   198 ; Equated Symbols:
                     0000   199 ;
                     0000   200 ;
                     0000   201 ; Own Storage:
                     0000   202 ;
                     0000   203 ; Table of XABs to be displayed after an indexed file is created.
                     0000   204 ;
                     0000   205 XABCRE_ARGS:
00'0C 16             0000   206         .BYTE   XAB$C_SUM, XAB$C_SUMLEN, XBC$C_OPNSUM3
00'40 15             0003   207         .BYTE   XAB$C_KEY, XAB$C_KEYLEN_V2, XBC$C_OPNKEY3
00'20 14             0006   208         .BYTE   XAB$C_ALL, XAB$C_ALLLEN, XBC$C_OPNALL3
      00             0009   209         .BYTE   0                                               ; end of table
                     000A   210
                     000A   211 XCONN03_ARGS:
00'54 21             000A   212         .BYTE   XAB$C_CXR,XAB$C_CXRLEN,XBC$C_XCONN03
      00             000D   213         .BYTE   0                                               ; end of table
```

```
                     000E   215 ;++
                     000E   216 ;
                     000E   217 ; RM$ALLOC3 - this routine is a bliss/mars interface.  called from rm3allbkt.
                     000E   218 ;
                     000E   219 ; Calling sequence:
                     000E   220 ;       rm$alloc3()
                     000E   221 ;
                     000E   222 ; Input Parameters:
                     000E   223 ;       r7 - address of area descriptor
                     000E   224 ;       r10 - address of internal fab
                     000E   225 ;
                     000E   226 ; Implicit Inputs:
                     000E   227 ;       none
                     000E   228 ;
                     000E   229 ; Output Parameters:
                     000E   230 ;       r1 - start vbn
                     000E   231 ;       r2 - vbn plus one of last vbn allocated
                     000E   232 ;
                     000E   233 ; Implicit Outputs:
                     000E   234 ;       none
                     000E   235 ;
                     000E   236 ; Routine Value:
                     000E   237 ;       none
                     000E   238 ;
                     000E   239 ; Side Effects:
                     000E   240 ;       none
                     000E   241 ;
                     000E   242 ;--
                     000E   243
                     000E   244 RM$ALLOC3::
 0078 8F     BB      000E   245         PUSHR   #^M<R3,R4,R5,R6>        ; save registers that are clobbered by exten
                     0012   246
                     0012   247 ;
                     0012   248 ; allocate a fib to do extend
                     0012   249 ;
                     0012   250
       52    40 8F   9A  0012   251         MOVZBL  #FIB$C_LENGTH,R2        ; size of fib
             FFE7'     30  0016   252         BSBW    RM$GETSPC1              ; get space for fib
             36 50     E9  0019   253         BLBC    R0,20$                  ; if error, get out
                           001C   254
                           001C   255 ;
                           001C   256 ; if the deq(area_desc) is zero then default to the higher of rtde(ifab) and
                           001C   257 ; arbktsz(area_desc).  in any case, allocate an even number of buckets.
                           001C   258 ;
                           001C   259
       50    03 A7   9A  001C   260         MOVZBL  AREA$B_ARBKTSZ(R7),R0   ; # of vbn's to a bucket
       52    24 A7   3C  0020   261         MOVZWL  AREA$W_DEQ(R7),R2       ; area's default extend quantity
                04     12  0024   262         BNEQ    10$                     ; if it doesn't exists,
       52    4C AA   3C  0026   263         MOVZWL  IFB$W_RTDEQ(R10),R2     ; use default run time quantity instead
          50    52     D1  002A   264 10$:    CMPL    R2,R0                   ; compare extend quantity to bucket size
                03     1E  002D   265         BGEQU   15$                     ; if less than one bucket
          52    50     D0  002F   266         MOVL    R0,R2                   ; use bucket size as extend quntity
                53     D4  0032   267 15$:    CLRL    R3                      ; quotient is quad word
 53   5C  52    50     7B  0034   268         EDIV    R0,R2,AP,R3             ; check if extend quantity is even multiple
                53     D5  0039   269         TSTL    R3                      ; if remainder is non_zero,
                06     13  003B   270         BEQL    18$                     ; then not even multiple
          50    53     C2  003D   271         SUBL2   R3,R0                   ; calc number to add to reach even multiple
```

```
                52    50    C0  0040    272             ADDL2   R0,R2                           ; make extend quantity even multiple
             18 A1    52    D0  0043    273 18$:        MOVL    R2,FIB$L_EXSZ(R1)               ; store quantity in fib
          28 A1    0C A7    D0  0047    274             MOVL    AREA$L_CVBN(R7),FIB$L_LOC_ADDR(R1) ; locate extend near last one
                FFB1'  30    004C    275             BSBW    RM$EXTEND0_ALT                  ; do extend
                52    56    D0  004F    276             MOVL    R6,R2                           ; change output register
                0078 8F     BA  0052    277 20$:        POPR    #^M<R3,R4,R5,R6>
                            05  0056    278             RSB
                                0057    279
                                0057    280
```

```
                                0057   282
                                0057   283 ;++
                                0057   284 ;
                                0057   285 ; RM$CONNECT3 - interfaces between rms0connect and rm3connect
                                0057   286 ;
                                0057   287 ; Calling sequence:
                                0057   288 ;       entered via a case statement (branch) from rms$connect
                                0057   289 ;
                                0057   290 ; Input Parameters:
                                0057   291 ;       none
                                0057   292 ;
                                0057   293 ; Implicit Inputs:
                                0057   294 ;       none
                                0057   295 ;
                                0057   296 ; Output Parameters:
                                0057   297 ;       none
                                0057   298 ;
                                0057   299 ; Implicit Outputs:
                                0057   300 ;       none
                                0057   301 ;
                                0057   302 ; Routine Value:
                                0057   303 ;       none
                                0057   304 ;
                                0057   305 ; Side Effects:
                                0057   306 ;       Mostly just passing through.
                                0057   307 ;
                                0057   308 ;       If this is a restart connect operation, then call XAB_SCAN which
                                0057   309 ;       will call XCONN03 to copy all Next Record Positioning information
                                0057   310 ;       into the keybuffers and IRAB for restarting.
                                0057   311 ;
                                0057   312 ;--
                                0057   313
                                0057   314 RM$CONNECT3::
00000000'EF   16                0057   315         JSB     RM$CONNECT3B
      29 50   E9                005D   316         BLBC    R0,10$
   22 69   3B   E1              0060   317         BBC     #IRB$V_RESTART,(R9),5$  ; If not restarting, just exit
           5C   DD              0064   318         PUSHL   AP                      ; Save ap - needed by XAB_SCAN
   5C   A1 AF   9E              0066   319         MOVAB   XCONN03_ARGS,AP         ; arguments to XAB_SCAN
         FF93'  30              006A   320         BSBW    RM$XAB_SCAN             ; call XAB_SCAN to copy NRP info
           5C 8ED0              006D   321         POPL    AP                      ; restore the ap
      13 50   E8                0070   322         BLBS    R0,5$                   ; restart was a success
    0000'CF   9F                0073   323         PUSHAB  W^RM$EX_NIRAB_SHR       ; return PC to STACK (i.e., to fake a BSBW)
   0E 69   22   E1              0077   324         BBC     #IRB$V_PPF_IMAGE,(R9),10$       ; branch if not INDIRECT PPF
6E   0000'CF   9E              007B   325         MOVAB   W^RM$EXRMS,(SP)         ; take structured exit
         FF7D'  30              0080   326         BSBW    RM$DISCONNECT3B         ; deallocate all the ISAM related buffers
         FF7A'  31              0083   327         BRW     RM$DISCOMMON            ; deallocate the IRAB and rest of stuff
                                0086   328                                         ; and exit with error (status in R0)
         FF77'  31              0086   329 5$:     BRW     RM$EXSUC                ; exit w/ success
         FF74'  31              0089   330 10$:    BRW     RM$EX_NIRAB_SHR         ; exit on error
```

```
                                ;++
                                ;
                                ; RMSCREATE3 - interfaces between rms0create and rm3create
                                ;
                                ; Calling sequence:
                                ;       entered via a case statement (branch) from rm$create
                                ;
                                ; Input Parameters:
                                ;       none
                                ;
                                ; Implicit Inputs:
                                ;       none
                                ;
                                ; Output Parameters:
                                ;       none
                                ;
                                ; Implicit Outputs:
                                ;       none
                                ;
                                ; Routine Value:
                                ;       none
                                ;
                                ; Side Effects:
                                ;       just passing through
                                ;
                                ;--

                                RM$CREATE3::

                                ;
                                ; do random device test here where the device characteristics bits can be defined
                                ;

            05   E0  008C      BBS     #IFB$V_BIO,-
        04 22 A9        008E            IFB$B_FAC(R9),4$        ; any device is ok for bio access
            1C   E1  0091      BBC     #DEV$V_RND,-
          35 69          0093            IFB$L_PRIM_DEV(R9),20$
         00AC C9   D4  0095  4$:     CLRL    IFB$L_IDX_PTR(R9)       ; clear pointer to index(may not be done yet
  00000000'EF      16  0099      JSB     RM$CREATE3B
05 04 A8      19  E1  009F      BBC     #FAB$V_CIF,FAB$L_FOP(R8),5$
         50   01  D1  00A4      CMPL    #1,R0                   ; cif and normal = open
              2C  13  00A7      BEQL    30$
           10 50  E9  00A9  5$:  BLBC    R0,10$                  ; if error from create, branch
              50  DD  00AC      PUSHL   R0                      ; save success code returned from create
              1C  E1  00AE      BBC     #DEV$V_RND,-            ; if not disk, do not do display
            12 69         00B0            IFB$L_PRIM_DEV(R9),15$
 5C     FF4A CF   9E  00B2      MOVAB   XABCRE_ARGS,AP          ; prepare table address for display
          FF46'    30  00B7      BSBW    RM$XAB_SCAN             ; do display of sum, key, and all XABs
              08  11  00BA      BRB     15$                     ; exit with success from create
              50  DD  00BC  10$:  PUSHL   R0                      ; push error code in stack
         5A   59   D0  00BE      MOVL    R9,R10                  ; setup shared ifab for return of space in c
          FF3C'    30  00C1      BSBW    RM$CLOSE3               ; give back key descriptors
  00000000'EF      17  00C4  15$:  JMP     RM$CREATEXIT1           ; R0 already in stack
                       00CA  20$:  RMSERR  DEV
  00000000'EF      17  00CF      JMP     RM$CLSCU
```

```
                00000000'EF   17  00D5   389 30$:    JMP     RM$OPEN_CIF
                              00DB   390
```

```
                     00DB   392 ;++
                     00DB   393 ;
                     00DB   394 ; RM$DELETE3 - this routine interfaces common delete
                     00DB   395 ;       with delete for indexed file org
                     00DB   396 ;
                     00DB   397 ; Calling sequence:
                     00DB   398 ;       entered via a case statement (branch) from rms$delete
                     00DB   399 ;
                     00DB   400 ; Input Parameters:
                     00DB   401 ;       none
                     00DB   402 ;
                     00DB   403 ; Implicit Inputs:
                     00DB   404 ;       none
                     00DB   405 ;
                     00DB   406 ; Output Parameters:
                     00DB   407 ;       none
                     00DB   408 ;
                     00DB   409 ; Implicit Outputs:
                     00DB   410 ;       none
                     00DB   411 ;
                     00DB   412 ; Routine Value:
                     00DB   413 ;       none
                     00DB   414 ;
                     00DB   415 ; Side Effects:
                     00DB   416 ;       just passing through
                     00DB   417 ;
                     00DB   418 ;--
                     00DB   419
                     00DB   420 RM$DELETE3::
          00   E1   00DB   421         BBC     #IFB$V_RU_RECVR,-       ; if the current operation has been
 0B 00A1   CA        00DD   422                 IFB$B_RECVRFLGS(R10),5$ ; done within the context of a Recovery
    7E     05   9A   00E1   423         MOVZBL  #RJR$_DELETE,-(SP)      ; Unit ROLLBACK, then call the common
00000000'EF   16   00E4   424         JSB     RM$RU_RECOVER           ; RU ROLLBACK routine to direct the
          12   11   00EA   425         BRB     20$                     ; UNDO operation, and then go finish up
                     00EC   426
 00 69    26   E4   00EC   427 5$:     BBSC    #IRB$V_PUTS_LAST,(R9),10$ ; clear the last operation
                     00F0   428                                         ; was a put sequential flag.
00000000'EF   16   00F0   429 10$:    JSB     RM$DELETE3B
    00B0 C9   D4   00F6   430         CLRL    IRB$L_UDR_VBN(R9)
    00BC C9   B4   00FA   431         CLRW    IRB$W_UDR_ID(R9)
      10 A8   D4   00FE   432 20$:    CLRL    RAB$W_RFA(R8)
      14 A8   B4   0101   433         CLRW    RAB$W_RFA+4(R8)
       FEF9'  31   0104   434         BRW     RM$EXRMS
                     0107   435
```

```
                     0107   437
                     0107   438 ;++
                     0107   439 ;
                     0107   440 ; RM$DISCONNECT3 - interfaces between rms0disconnect and rm3disconnect
                     0107   441 ;
                     0107   442 ; Calling sequence:
                     0107   443 ;       entered via a case statement (branch) from rms$disconnect
                     0107   444 ;
                     0107   445 ; Input Parameters:
                     0107   446 ;       none
                     0107   447 ;
                     0107   448 ; Implicit Inputs:
                     0107   449 ;       none
                     0107   450 ;
                     0107   451 ; Output Parameters:
                     0107   452 ;       none
                     0107   453 ;
                     0107   454 ; Implicit Outputs:
                     0107   455 ;       none
                     0107   456 ;
                     0107   457 ; Routine Value:
                     0107   458 ;       none
                     0107   459 ;
                     0107   460 ; Side Effects:
                     0107   461 ;       just passing through
                     0107   462 ;
                     0107   463 ;--
                     0107   464
                     0107   465 RM$DISCONNECT3::
FEF6'  30            0107   466         BSBW    RM$DISCONNECT3B
FEF3'  31            010A   467         BRW     RM$DISCOMMON            ; branch to common code to finish up
                     010D   468
```

```
                     010D   470
                     010D   471 ;++
                     010D   472 ;
                     010D   473 ; RM$EXTEND3 - extend for isam files
                     010D   474 ;
                     010D   475 ; Calling sequence:
                     010D   476 ;       entered via a case statement (branch) from rms$extend
                     010D   477 ;
                     010D   478 ; Input Parameters:
                     010D   479 ;       r3      - address of fab/xab
                     010D   480 ;       r5      - extend quantity
                     010D   481 ;       r6      - 0 if no xab otherwise xab address
                     010D   482 ;
                     010D   483 ; Implicit Inputs:
                     010D   484 ;       none
                     010D   485 ;
                     010D   486 ; Output Parameters:
                     010D   487 ;       none
                     010D   488 ;
                     010D   489 ; Implicit Outputs:
                     010D   490 ;       none
                     010D   491 ;
                     010D   492 ; Routine Value:
                     010D   493 ;       none
                     010D   494 ;
                     010D   495 ; Side Effects:
                     010D   496 ;       just passing through
                     010D   497 ;
                     010D   498 ;--
                     010D   499
                     010D   500 RM$EXTEND3::
          08   BB   010D   501         PUSHR   #^M<R3>                 ; push fab/xab addr since clobbered by rm$ex
          7E   D4   010F   502         CLRL    -(SP)                   ; area zero is default
          56   D5   0111   503         TSTL    R6                      ; is there an xab
          04   13   0113   504         BEQL    10$                     ; if not use area default
     6E   17 A3   D0   0115   505         MOVL    XAB$B_AID(R3),(SP)      ; pickup area out of xab
                     0119   506
        FEE4'  30   0119   507 10$:     BSBW    RM$LOCK_AREA            ; returns bdb=r4, area_desc=r7
  0C A8    8E   D0   011C   508         MOVL    (SP)+,FAB$L_STV(R8)     ; save level for next error check
     6E 50   E9   0120   509         BLBC    R0,50$                  ; if errors, get out
  53   6E   D0   0123   510         MOVL    (SP),R3                 ; restore fab/xab address
          54   DD   0126   511         PUSHL   R4                      ; save bdb
                     0128   512
        1C A7   D5   0128   513         TSTL    AREA$L_NXT(R7)          ; check if next extent is unde'ined
          50   12   012B   514         BNEQ    30$                     ; if not, error
        0C A8   D4   012D   515         CLRL    FAB$L_STV(R8)           ; no stv value
        FECD'  30   0130   516         BSBW    RM$EXTEND0              ; do extend, cloobers r4
     4C 50   E9   0133   517         BLBC    R0,40$                  ; check for errors and get out
  74 A9    56   D0   0136   518         MOVL    R6,IFB$L_EBK(R9)        ; update eof
  53   04 AE   D0   013A   519         MOVL    4(SP),R3                ; restore fab/xab address
10 A3   56   51   C3   013E   520         SUBL3   R1,R6,FAB$L_ALQ(R3)     ; calc number of blocks in extent
                     0143   521 ;
                     0143   522 ; To keep the total allocation correct, we must distinguish between files
                     0143   523 ; that did not have this field defined when they were created, and those
                     0143   524 ; that did.  If the TOTAL_ALLOC field is not zero, then it is a new file.
                     0143   525 ; If it is zero, check the VBN of the extents.  If they are both zero, then
                     0143   526 ; the area has never been extended, and the allocation can be computed
```

```
                                0143   527 ; correctly.
                                0143   528 ;
                             D5 0143   529          TSTL    AREA$L_TOTAL_ALLOC(R7)  ; is total allocation zero?
                             12 0146   530          BNEQ    15$                     ; if not, new file
                        0C A7   D5 0148   531          TSTL    AREA$L_CVBN(R7)         ; is current extend zero?
                             0A 12 014B   532          BNEQ    16$                     ; if not, old file
                        18 A7   D5 014D   533          TSTL    AREA$L_NXTVBN(R7)       ; is next extent zero?
                             05 12 0150   534          BNEQ    16$                     ; if not, old file
                        10 A3   C0 0152   535 15$.     ADDL2   FAB$L_ALQ(R3),-
                        32 A7      0155   536                  AREA$L_TOTAL_ALLOC(R7) ; new type file, accum total alloc
                0C A8   10 A3   C0 0157   537 16$:     ADDL2   FAB$L_ALQ(R3),FAB$L_STV(R8) ; accumulate in fab
                   1C A7   51   D0 015C   538          MOVL    R1,AREA$L_NXT(R7)       ; save start vbn
                20 A7   10 A3   D0 0160   539          MOVL    FAB$L_ALQ(R3),AREA$L_NXBLK(R7) ; remember # of blocks in extent
                                0165   540
                          54 8ED0 0165   541          POPL    R4                      ; restore bdb of area prologue vbn
                      55   18 A4   D0 0168   542          MOVL    BDB$L_ADDR(R4),R5       ; pickup address of area descriptor
                           FE91'   30 016C   543          BSBW    RM$MARSUM               ; calc check sum
                        0A A4   03   88 016F   544          BISB2   #3,BDB$B_FLGS(R4)       ; set valid and dirty
                           53   02   D0 0173   545          MOVL    #RLS$M_WRT_THRU,R3      ; write it out
                           FE87'   30 0176   546          BSBW    RM$RELEASE
                             53 8ED0 0179   547 20$:     POPL    R3
                                05 017C   548          RSB
                                017D   549 30$:     RMSERR  LEX                     ; set error code for next extend already exi
                             54 8ED0 0182   550 40$:     POPL    R4
                             50   DD 0185   551          PUSHL   R0
                             53   D4 0187   552          CLRL    R3                      ; just release
                           FE74'   30 0189   553          BSBW    RM$RELEASE              ; do it
                             50 8ED0 018C   554          POPL    R0                      ; pickup saved error code
                             E8   11 018F   555          BRB     20$
                        0C A8   6E   D0 0191   556 50$:     MOVL    (SP),FAB$L_STV(R8)      ; stv value for error is fab/xab
                             E2   11 0195   557          BRB     20$
                                0197   558
```

```
                     0197   560
                     0197   561 ;++
                     0197   562 ;
                     0197   563 ; RM$GET3 - interface to RM$GET3B
                     0197   564 ; RM$FIND3 - interface to RM$GET3B for find operation
                     0197   565 ;
                     0197   566 ; Calling sequence:
                     0197   567 ;       entered via a case statement (branch) from rms$get (find)
                     0197   568 ;
                     0197   569 ; Input Parameters:
                     0197   570 ;       none
                     0197   571 ;
                     0197   572 ; Implicit Inputs:
                     0197   573 ;       none
                     0197   574 ;
                     0197   575 ; Output Parameters:
                     0197   576 ;       none
                     0197   577 ;
                     0197   578 ; Implicit Outputs:
                     0197   579 ;       none
                     0197   580 ;
                     0197   581 ; Routine Value:
                     0197   582 ;       none
                     0197   583 ;
                     0197   584 ; Side Effects:
                     0197   585 ;       just passing through
                     0197   586 ;
                     0197   587 ;--
                     0197   588
                     0197   589 RM$GET3::
 22 A8    B4         0197   590          CLRW    RAB$W_RSZ(R8)           ; init record size
                     019A   591 RM$FIND3::
    00    E1         019A   592          BBC     #IFB$V_RU_RECVR,-       ; if the current operation has been
 0B 00A1  CA         019C   593                  IFB$B_RECOVRFLGS(R10),5$ ; done within the context of a Recovery
    7E    0B    9A   01A0   594          MOVZBL  #RJR$_FIND,-(SP)        ; Unit ROLLBACK, then call the common
00000000'EF   16    01A3   595          JSB     RM$RU_RECOVER           ; RU ROLLBACK routine to direct the
    0B    11        01A9   596          BRB     20$                     ; UNDO operation, and then go finish up
                     01AB   597
    26    E4        01AB   598 5$:      BBSC    #IRB$V_PUTS_LAST,-      ; clear the last operation was a
 00 69              01AD   599                  (R9),7$                 ; sequential $PUT flag
 00 69    33    E4   01AF   600 7$:      BBSC    #IRB$V_UPDATE,(R9),10$  ; clear last operation was $UPDATE flag
                     01B3   601
  FE4A'    30        01B3   602 10$:     BSBW    RM$GET3B                ; get/find the record
  FE47'    31        01B6   603 20$:     BRW     RM$EXRMS                ; and exit rms
                     01B9   604
```

```
                     01B9   606
                     01B9   607 ;++
                     01B9   608 ;
                     01B9   609 ; RM$IALLO3 - interface xab_scan to the routine which processes the allocation xab
                     01B9   610 ; RM$IKEYO3 - interface xab_scan to the routine which processes the key xab.
                     01B9   611 ; RM$ISUMO3 - interface xab_scan to the routine which processes the summary xab.
                     01B9   612 ;
                     01B9   613 ; Calling sequence:
                     01B9   614 ;       entered via a case statement (branch) from rm$xab_scan
                     01B9   615 ;
                     01B9   616 ; Input Parameters:
                     01B9   617 ;       none
                     01B9   618 ;
                     01B9   619 ; Implicit Inputs:
                     01B9   620 ;       none
                     01B9   621 ;
                     01B9   622 ; Output Parameters:
                     01B9   623 ;       none
                     01B9   624 ;
                     01B9   625 ; Implicit Outputs:
                     01B9   626 ;       none
                     01B9   627 ;
                     01B9   628 ; Routine Value:
                     01B9   629 ;       none
                     01B9   630 ;
                     01B9   631 ; Side Effects:
                     01B9   632 ;       just passing through
                     01B9   633 ;
                     01B9   634 ;--
                     01B9   635 ;
                     01B9   636
                     01B9   637 ;++
                     01B9   638 ;
                     01B9   639 ; RM$IALLO3 - interface xab_scan to the routine which processes the
                     01B9   640 ;       allocation xab's for index seq files.
                     01B9   641 ;--
                     01B9   642
                     01B9   643 RM$IALLO3::
     52   5C    D0   01B9   644          MOVL    AP,R2                   ; save ap
          53    DD   01BC   645          PUSHL   R3
00000000'EF    16   01BE   646          JSB     RM$XALLO3
     5E   04    C0   01C4   647          ADDL2   #4,SP
          1E    11   01C7   648          BRB     ISUM10
                     01C9   649
                     01C9   650 ;--
                     01C9   651 ;++
                     01C9   652 ;
                     01C9   653 ; RM$IKEYO3 - interface xab_scan to the routine which processes the key xab.
                     01C9   654 ;
                     01C9   655 ;--
                     01C9   656
                     01C9   657 RM$IKEYO3::
     52   5C    D0   01C9   658          MOVL    AP,R2                   ; save ap
          53    DD   01CC   659          PUSHL   R3
00000000'EF    16   01CE   660          JSB     RM$XKEYO3
     5E   04    C0   01D4   661          ADDL2   #4,SP
          0E    11   01D7   662          BRB     ISUM10
```

```
                     01D9   663
                     01D9   664 ;++
                     01D9   665 ;
                     01D9   666 ; RM$ISUM03 - interface xab_scan to the routine which processes the summary xab.
                     01D9   667 ;
                     01D9   668 ;--
                     01D9   669
                     01D9   670 RM$ISUM03::
    52    5C    D0   01D9   671          MOVL    AP,R2                   ; save ap
          18    BB   01DC   672          PUSHR   #^M<R3,R4>              ; push parameters
00000000'EF     16   01DE   673          JSB     RM$XSUM03
    5E    08    C0   01E4   674          ADDL2   #8,SP
      06  50    E8   01E7   675 ISUM10:  BLBS    R0,ISUM20
 0C A8    53    D0   01EA   676          MOVL    R3,FAB$L_STV(R8)
          8E    D5   01EE   677          TSTL    (SP)+
    5C    52    D0   01F0   678 ISUM20:  MOVL    R2,AP                   ; restore ap
                05   01F3   679          RSB
                     01F4   680
                     01F4   681
```

```
                     01F4   683
                     01F4   684 ;++
                     01F4   685 ;
                     01F4   686 ; RM$OPEN3 - interfaces between rms0open and rm3open
                     01F4   687 ;
                     01F4   688 ; Calling sequence:
                     01F4   689 ;       entered via a case statement (branch) from rm$open
                     01F4   690 ;
                     01F4   691 ; Input Parameters:
                     01F4   692 ;       none
                     01F4   693 ;
                     01F4   694 ; Implicit Inputs:
                     01F4   695 ;       none
                     01F4   696 ;
                     01F4   697 ; Output Parameters:
                     01F4   698 ;       none
                     01F4   699 ;
                     01F4   700 ; Implicit Outputs:
                     01F4   701 ;       none
                     01F4   702 ;
                     01F4   703 ; Routine Value:
                     01F4   704 ;       none
                     01F4   705 ;
                     01F4   706 ; Side Effects:
                     01F4   707 ;       just passing through
                     01F4   708 ;
                     01F4   709 ;--
                     01F4   710
                     01F4   711 RM$OPEN3::
          FE09'  30  01F4   712          BSBW    RM$OPEN3B
        08 50    E8  01F7   713          BLBS    R0,10$
           50    DD  01FA   714          PUSHL   R0
          FE01'  30  01FC   715          BSBW    RM$CLOSE3
           50 8ED0   01FF   716          POPL    R0
00000000'EF    17   0202   717 10$:     JMP     RM$COPRTN
                    0208   718
```

```
                     0208   720
                     0208   721 ;++
                     0208   722 ;
                     0208   723 ; RM$PUT3 - interfaces between rms0put and rm3put
                     0208   724 ;
                     0208   725 ; Calling sequence:
                     0208   726 ;       entered via a case statement (branch) from rms$put
                     0208   727 ;
                     0208   728 ; Input Parameters:
                     0208   729 ;       none
                     0208   730 ;
                     0208   731 ; Implicit Inputs:
                     0208   732 ;       none
                     0208   733 ;
                     0208   734 ; Output Parameters:
                     0208   735 ;       none
                     0208   736 ;
                     0208   737 ; Implicit Outputs:
                     0208   738 ;       none
                     0208   739 ;
                     0208   740 ; Routine Value:
                     0208   741 ;       none
                     0208   742 ;
                     0208   743 ; Side Effects:
                     0208   744 ;       just passing through
                     0208   745 ;
                     0208   746 ;--
                     0208   747
                     0208   748 RM$PUT3::
          00   E1   0208   749         BBC     #IFB$V_RU_RECVR,-       ; if the current operation has been
    0B 00A1 CA       020A   750                 IFB$B_RECVRFLGS(R10),10$; done within the context of a Recovery
       7E   13   9A  020E   751         MOVZBL  #RJR$_PUT,-(SP)         ; Unit ROLLBACK, then call the common
 00000000'EF   16   0211   752         JSB     RM$RU_RECOVER           ; RU ROLLBACK routine to direct the
          2/   11   0217   753         BRB     PUF2                    ; UNDO operation, and then go finish up
                     0219   754
        FDE4'  30   0219   755 10$:    BSBW    RM$PUT3B                ; put the record
          50   DD   021C   756         PUSHL   R0                      ; save status
       05 50   E9   021E   757         BLBC    R0,UNLOCK2              ; always unlock on errors
                     0221   758
10 04 A8   12   E0   0221   759 UNLOCK: BBS     #RAB$V_ULK,RAB$L_ROP(R8),PUF; don't unlock if manual unlock
                     0226   760
                     0226   761 UNLOCK2:
       69   2D   E5   0226   762         BBCC    #IRB$V_UNLOCK_RP,(R9),- ; if there isn't any record to unlock
          0C       0229   763                 PUF                     ; then don't unlock any record
  51    78 A9   D0   022A   764         MOVL    IRB$L_PUTUP_VBN(R9),R1  ; get vbn
 52  0080 C9   3C   022E   765         MOVZWL  IRB$W_PUTUP_ID(R9),R2   ; set up id for unlock
        FDCA'  30   0233   766         BSBW    RM$UNLOCK               ; unlock the record (ignore errors,
                     0236   767                                         ; record maybe not locked)
                     0236   768
       00B0 C9   D4   0236   769 PUF:     CLRL    IRB$L_UDR_VBN(R9)       ; Having attempted a $PUT/$UPDATE
       00BC C9   B4   023A   770         CLRW    IRB$W_UDR_ID(R9)        ; there is no longer a current record
          01   BA   023E   771         POPR    #^M<R0>                 ; restore status
          FDBD'  31   0240   772 PUF2:    BRW     RM$EXRMS                ; and exit
```

```
                     0243   774
                     0243   775 ;++
                     0243   776 ;
                     0243   777 ; RM$REWIND3 - isam rewind
                     0243   778 ;
                     0243   779 ; Calling sequence:
                     0243   780 ;       entered via a case statement (branch) from rms$rewind
                     0243   781 ;
                     0243   782 ; Input Parameters:
                     0243   783 ;       none
                     0243   784 ;
                     0243   785 ; Implicit Inputs:
                     0243   786 ;       none
                     0243   787 ;
                     0243   788 ; Outputs Parameters:
                     0243   789 ;       none
                     0243   790 ;
                     0243   791 ; Implicit Outputs:
                     0243   792 ;       none
                     0243   793 ;
                     0243   794 ; Routine Value:
                     0243   795 ;       none
                     0243   796 ;
                     0243   797 ; Side Effects:
                     0243   798 ;       just passing through
                     0243   799 ;
                     0243   800 ;--
                     0243   801 ;
                     0243   802 ;       this routine does the indexed specific code for a rewind
                     0243   803 ;               but it was just too small and much easier in mars
                     0243   804 ;               for it to go into another module
                     0243   805 ;--
                     0243   806
                     0243   807 RM$REWIND3::
     00 69   26  E4  0243   808         BBSC    #IRB$V_PUTS_LAST,(R9),5$ ; clear the last operation
                     0247   809                                          ; was a put sequential flag.
  05 22 AA   05  E1  0247   810 5$:     BBC     #IFB$V_BIO,IFB$B_FAC(R10),10$ ; branch if not block i/o
     40 A9   01  D0  024C   811         MOVL    #1,IRB$L_NRP_VBN(R9)
                 05  0250   812         RSB                                  ; all done if block i/o
                     0251   813
                     0251   814 ;
                     0251   815 ; Clear the next record positioning context information within the IRAB,
                     0251   816 ; and setup the key of reference from information in the RAB.
                     0251   817 ;
                     0251   818
                     0251   819         ASSUME  IRB$L_CUR_VBN+4  EQ IRB$L_POS_VBN
                     0251   820         ASSUME  IRB$L_UDR_VBN+4  EQ IRB$L_SIDR_VBN
                     0251   821         ASSUME  IRB$W_CUR_ID+2   EQ IRB$W_POS_ID
                     0251   822         ASSUME  IRB$W_CUR_ID+4   EQ IRB$W_UDR_ID
                     0251   823         ASSUME  IRB$W_CUR_ID+6   EQ IRB$W_SIDR_ID
        00A8 C9  7C  0251   824 10$:     CLRQ    IRB$L_CUR_VBN(R9)
        00B0 C9  7C  0255   825         CLRQ    IRB$L_UDR_VBN(R9)
        00B8 C9  7C  0259   826         CLRQ    IRB$W_CUR_ID(R9)
        00C0 C9  B4  025D   827         CLRW    IRB$W_CUR_COUNT(R9)
00C3 C9    35 A8  90  0261   828         MOVB    RAB$B_KRF(R8),IRB$B_CUR_KREF(R9)
                     0267   829
                     0267   830 ;
```

```
                                 0267   831 ; This stream can no lnger be explicitely positioned at the end-of-file
                                 0267   832 ;
                                 0267   833
                                 0267   834          CSB     #IRB$V_EOF,(R9)
                                 026B   835
                                 026B   836 ;
                                 026B   837 ; zero out keybuffers 1 and 2
                                 026B   838 ;
                                 026B   839
                50   00B4 CA    3C   026B   840          MOVZWL  IFB$W_KBUFSZ(R10),R0
                50   50   01    78   0270   841          ASHL    #1,R0,R0
    60 B9       50   00   6E    00   2C   0274   842          MOVC5   #0,(SP),#0,R0,@IRB$L_KEYBUF(R9)
                                      05   027B   843          RSB
                                           027C   844
```

```
                     027C    846
                     027C    847 ;++
                     027C    848 ;
                     027C    849 ; RM$RND_DEV
                     027C    850 ;       This routine is called from RM$CREATE3B to determine if the
                     027C    851 ;       indexed file is being created in a random access, non foreign device.
                     027C    852 ;       It has to be done in a macro routine, since it is the only way
                     027C    853 ;       the device characteristics bits can be defined.
                     027C    854 ;
                     027C    855 ; Calling sequence:
                     027C    856 ;       RM$RND_DEV ()
                     027C    857 ;
                     027C    858 ; Input Parameters:
                     027C    859 ;       IFAB
                     027C    860 ;
                     027C    861 ; Implicit Inputs:
                     027C    862 ;       none
                     027C    863 ;
                     027C    864 ; Output Parameters:
                     027C    865 ;       none
                     027C    866 ;
                     027C    867 ; Implicit Outputs:
                     027C    868 ;       none
                     027C    869 ;
                     027C    870 ; Routine Value:
                     027C    871 ;       0 - not a random device
                     027C    872 ;       1 - random device
                     027C    873 ;
                     027C    874 ; Side Effects:
                     027C    875 ;
                     027C    876 ;--
                     027C    877
                     027C    878 RM$RND_DEV::
                     027C    879
      50    D4       027C    880          CLRL    R0                              ; Assume device is not random
      11    E0       027E    881          BBS     #FAB$V_UFO,-
0B 04 A8             0280    882                   FAB$L_FOP(R8),10$               ; branch if UFO
      18    E0       0283    883          BBS     #DEV$V_FOR,-
   07 69             0285    884                   IFB$L_PRIM_DEV(R9),10$          ; branch if device foreign
      1C    E1       0287    885          BBC     #DEV$V_RND,-
   03 69             0289    886                   IFB$L_PRIM_DEV(R9),10$          ; branch if device not random
50    01    D0       028B    887          MOVL    #1,R0                           ; device is random
            05       028E    888 10$:     RSB                                     ; return
                     028F    889
```

```
                     028F    891
                     028F    892 ;++
                     028F    893 ;
                     028F    894 ; RM$UPDATE3 - interfaces between rms0update and rm3update
                     028F    895 ;
                     028F    896 ; Calling sequence:
                     028F    897 ;       entered via a case statement (branch) from rms$update
                     028F    898 ;
                     028F    899 ; Input Parameters:
                     028F    900 ;       none
                     028F    901 ;
                     028F    902 ; Implicit Inputs:
                     028F    903 ;       none
                     028F    904 ;
                     028F    905 ; Output Parameters:
                     028F    906 ;       none
                     028F    907 ;
                     028F    908 ; Implicit Outputs:
                     028F    909 ;       none
                     028F    910 ;
                     028F    911 ; Routine Value:
                     028F    912 ;       none
                     028F    913 ;
                     028F    914 ; Side Effects:
                     028F    915 ;       just passing through
                     028F    916 ;
                     028F    917 ;--
                     028F    918
                     028F    919 RM$UPDATE3::
          00   E1   028F    920         BBC     #IFB$V_RU_RECVR,-       ; if the current operation has been
    0B 00A1   CA     0291    921                 IFB$B_RECVRFLGS(R10),5$ ; done within the context of a Recovery
       7E   1C   9A 0295    922         MOVZBL  #RJR$_UPDATE,-(SP)      ; Unit ROLLBACK, then call the common
00000000'EF   16   0298    923         JSB     RM$RU_RECOVER           ; RU ROLLBACK routine to direct the
          0E   11   029E    924         BRB     15$                     ; UNDO operation, and then go finish up
                     02A0    925
                     02A0    926 5$:     SSB     #IRB$V_UPDATE,(R9)      ; set the update bit
    00 69   26   E4 02A4    927         BBSC    #IRB$V_PUTS_LAST,(R9),10$ ; clear the last operation
                     02A8    928                                         ; was a put sequential flag.
00000000'EF   16   02A8    929 10$:    JSB     RM$UPDATE3B             ; update the record
          50   DD   02AE    930 15$:    PUSHL   R0                      ; save status
        FF6E   31   02B0    931         BRW     UNLOCK                  ; go unlock the record if necessary
                     02B3    932         .END
```

Symbol table

```
$$.PSECT_EP                     = 00000000
$$RMSTEST                       = 0000001A
$$RMS_PBUGCHK                   = 00000010
$$RMS_TBUGCHK                   = 00000008
$$RMS_UMODE                     = 00000004
AREA$B_ARBKTSZ                  = 00000003
AREA$L_CVBN                     = 0000000C
AREA$L_NXBLK                    = 00000020
AREA$L_NXT                      = 0000001C
AREA$L_NXTVBN                   = 00000018
AREA$L_TOTAL_ALLOC              = 00000032
AREA$W_DEQ                      = 00000024
BDB$B_FLGS                      = 0000000A
BDB$L_ADDR                      = 00000018
DEV$V_FOR                       = 00000018
DEV$V_RND                       = 0000001C
FAB$L_ALQ                       = 00000010
FAB$L_FOP                       = 00000004
FAB$L_STV                       = 0000000C
FAB$V_CIF                       = 00000019
FAB$V_UFO                       = 00000011
FIB$C_LENGTH                    = 00000040
FIB$L_EXSZ                      = 00000018
FIB$L_LOC_ADDR                  = 00000028
IFB$B_FAC                       = 00000022
IFB$B_RECVRFLGS                 = 000000A1
IFB$L_EBK                       = 00000074
IFB$L_IDX_PTR                   = 000000AC
IFB$L_PRIM_DEV                  = 00000000
IFB$V_BIO                       = 00000005
IFB$V_RU_RECVR                  = 00000000
IFB$W_KBUFSZ                    = 000000B4
IFB$W_RTDEQ                     = 0000004C
IRB$B_CUR_KREF                  = 000000C3
IRB$L_CUR_VBN                   = 000000A8
IRB$L_KEYBUF                    = 00000060
IRB$L_NRP_VBN                   = 00000040
IRB$L_POS_VBN                   = 000000AC
IRB$L_PUTOP_VBN                 = 00000078
IRB$L_SIDR_VBN                  = 000000B4
IRB$L_UDR_VBN                   = 000000B0
IRB$V_EOF                       = 00000021
IRB$V_PPF_IMAGE                 = 00000022
IRB$V_PUTS_LAST                 = 00000026
IRB$V_RESTART                   = 0000003B
IRB$V_UNLOCK_RP                 = 0000002D
IRB$V_UPDATE                    = 00000033
IRB$W_CUR_COUNT                 = 000000C0
IRB$W_CUR_ID                    = 000000B8
IRB$W_POS_ID                    = 000000BA
IRB$W_PUTOP_ID                  = 00000080
IRB$W_SIDR_ID                   = 000000BE
IRB$W_UDR_ID                    = 000000BC
ISUM10                            000001E7 R      01
ISUM20                            000001F0 R      01
PUF                               00000236 R      01
PUF2                              00000240 R      01
RAB$B_KRF                       = 00000035
RAB$L_ROP                       = 00000004
RAB$V_ULK                       = 00000012
RAB$W_RFA                       = 00000010
RAB$W_RSZ                       = 00000022
RJR$_DELETE                     = 00000005
RJR$_FIND                       = 0000000B
RJR$_PUT                        = 00000013
RJR$_UPDATE                     = 0000001C
RLS$M_WRT_THRU                  = 00000002
RM$ALLOC3                         0000000E RG     01
RM$CLOSE3                         ********   X    01
RM$CLSCU                          ********   X    01
RM$CONNECT3                       00000057 RG     01
RM$CONNECT3B                      ********   X    01
RM$COPRTN                         ********   X    01
RM$CREATE3                        0000008C RG     01
RM$CREATE3B                       ********   X    01
RM$CREATEXIT1                     ********   X    01
RM$DELETE3                        000000DB RG     01
RM$DELETE3B                       ********   X    01
RM$DISCOMMON                      ********   X    01
RM$DISCONNECT3                    00000107 RG     01
RM$DISCONNECT3B                   ********   X    01
RM$EXRMS                          ********   X    01
RM$EXSUC                          ********   X    01
RM$EXTEND0                        ********   X    01
RM$EXTEND0_ALT                    ********   X    01
RM$EXTEND3                        0000010D RG     01
RM$EX_NIRAB_SHR                   ********   X    01
RM$FIND3                          0000019A RG     01
RM$GET3                           00000197 RG     01
RM$GET3B                          ********   X    01
RM$GETSPC1                        ********   X    01
RM$IALLO3                         000001B9 RG     01
RM$IKEYO3                         000001C9 RG     01
RM$ISUMO3                         000001D9 RG     01
RM$LOCK_AREA                      ********   X    01
RM$MAKSOM                         ********   X    01
RM$OPEN3                          000001F4 RG     01
RM$OPEN3B                         ********   X    01
RM$OPEN_CIF                       ********   X    01
RM$PUT3                           00000208 RG     01
RM$PUT3B                          ********   X    01
RM$RELEASE                        ********   X    01
RM$REWIND3                        00000243 RG     01
RM$RND_DEV                        0000027C RG     01
RM$RU_RECOVER                     ********   X    01
RM$UNLOCK                         ********   X    01
RM$UPDATE3                        0000028F RG     01
RM$UPDATE3B                       ********   X    01
RM$XAB_SCAN                       ********   X    01
RM$XALLO3                         ********   X    01
RM$XKEYO3                         ********   X    01
RM$XSUMO3                         ********   X    01
RMS$_DEV                        = 00184C4
RMS$_LEX                        = 0001878C
```

```
UNLOCK                          00000221 R      01
UNLOCK2                         00000226 R      01
XAB$B_AID                     = 00000017
XAB$C_ALL                     = 00000014
XAB$C_ALLLEN                  = 00000020
XAB$C_CXR                     = 00000021
XAB$C_CXRLEN                  = 00000054
XAB$C_KEY                     = 00000015
XAB$C_KEYLEN_V2               = 00000040
XAB$C_SUM                     = 00000016
XAB$C_SUMLEN                  = 0000000C
XABCRE_ARGS                     00000000 R      01
XBC$C_OPNALL3                   ********   X    01
XBC$C_OPNKEY3                   ********   X    01
XBC$C_OPNSUM3                   ********   X    01
XBC$C_XCONN03                   ********   X    01
XCONN03_ARGS                    0000000A R      01
```

+----------------+
! Psect synopsis !
+----------------+

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
RM$RMSOFACE	000002B3 (691.)	01 (1.)	PIC USR CON REL GBL NOSHR EXE RD NOWRT NOVEC LONG
ABS	00000000 (0.)	02 (2.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE

+-------------------------+
! Performance indicators !
+-------------------------+

Phase	Page faults	CPU Time	Elapsed Time
Initialization	36	00:00:00.07	00:00:01.07
Command processing	138	00:00:00.72	00:00:05.51
Pass 1	409	00:00:15.25	00:00:46.72
Symbol table sort	0	00:00:02.21	00:00:04.10
Pass 2	163	00:00:03.62	00:00:11.59
Symbol table output	16	00:00:00.16	00:00:00.75
Psect synopsis output	1	00:00:00.02	00:00:00.30
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	765	00:00:22.05	00:01:10.05

```
The working set limit was 1650 pages.
88691 bytes (174 pages) of virtual memory were used to buffer the intermediate code.
There were 90 pages of symbol table space allocated to hold 1644 non-local and 36 local symbols.
932 source lines were read in Pass 1, producing 16 object records in Pass 2.
30 pages of virtual memory were used to define 29 macros.
```

```
                                    +----------------------------+
                                    ! Macro library statistics !
                                    +----------------------------+

Macro library name                   Macros defined
------------------                   --------------
_$255$DUA28:[RMS.OBJ]RMS.MLB;1                19
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                 0
_$255$DUA28:[SYSLIB]STARLET.MLB;2              6
TOTALS (all libraries)                        25

1747 GLTS were required to define 25 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:RM3FACE/OBJ=OBJ$:RM3FACE MSRC$:RM3FACE/UPDATE=(ENH$:RM3FACE)+EXECML$/LIB+LIB$:RMS/LIB
```

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

RM3FACE
LIS

RM3DISCON
LIS

RM3CONN
LIS

RM3DELETE
LIS

RM3CREATE
LIS

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

RM3MAKIDX
LIS

RM3FNDRRV
LIS

RM3FNDRFA
LIS

RM3GET
LIS

RM3IUDR
LIS

RM3JOURNL
LIS

RM3KEYDSC
LIS

RM3MISC
LIS

RM3MISPUT
LIS